

1
2 In the Claims:

3 1. A multiple error correcting code (ECC) mechanism for use with transactions in
4 a computer system, comprising:

5 an ECC encoder, the ECC encoder applying a first ECC code of a first portion of
6 the transaction and a second ECC code to a second portion of the transaction; and

7 a decoder that decodes the first and the second ECC codes.

8 2. The mechanism of claim 1, wherein the first ECC code is a single error correction-
9 double error detection (SEC-DED) code.

10 3. The mechanism of claim 1, wherein the first portion of the transaction is a header
11 packet and wherein the second portion of the transaction comprises one or more data
12 packets.

13 4. The mechanism of claim 3, wherein the second ECC code is contained in a last
14 data packet of the one or more data packets.

15 5. The mechanism of claim 3, wherein the second ECC code comprises a plurality
16 of parity bits and one or more remaining ECC bits, wherein each data packet of the one
17 or more data packets includes one parity bit of the plurality of parity bits, wherein a last
18 data packet of the one or more data packets includes the one or more remaining data bits,
19 and wherein data may be processed without waiting for all of the one or packets to be
20 received if no parity bit errors exist.

21 6. The mechanism of claim 1, wherein the first portion of the transaction is a first
22 section of a packet and the second portion of the transaction is a second portion of the
23 packet.

24 7. The mechanism of claim 1, wherein the second ECC code is a SEC-DED code.

25 8. The mechanism of claim 1, wherein the second ECC code detects a wire failure.

26 9. The mechanism of claim 1, wherein the transaction further comprises a third
27 portion, and the mechanism further comprises a third ECC code.

28 10. A method for protecting a transaction in a computer system by using a multiple
29 error correcting code scheme, comprising:

30 applying a first ECC code to a first portion of the transaction;

31 applying a second ECC code to a second portion of the transaction;

1 transmitting the transaction; and
2 decoding the first and the second ECC codes.

3 11. The method of claim 10, wherein the first ECC code is a single error correction-
4 double error detection (SEC-DED) code.

5 12. The method of claim 10, wherein the first portion of the transaction is a header
6 packet and wherein the second portion of the transaction comprises one or more data
7 packets.

8 13. The method of claim 10, wherein the second portion of the transaction comprises
9 one or more data packets.

10 14. The method of claim 13, wherein the second ECC code is contained in a last data
11 packet of the one or more data packets.

12 15. The method of claim 13, wherein the second ECC code comprises a plurality of
13 parity bits and one or more remaining ECC bits, wherein each data packet of the one or
14 more data packets includes one parity bit of the plurality of parity bits, and wherein a last
15 data packet of the one or more data packets includes the one or more remaining data bits.

16 16. The method of claim 10, wherein the second ECC code is one of a SEC-DED
17 code and a code that detects a wire failure.

18 17. The method of claim 10, wherein the transaction further comprises a third portion,
19 and the mechanism further comprises a third ECC code.

20 18. The method of claim 10, further comprising:
21 defining all transactions and an amount of information to be protected;
22 defining interconnect widths and packet sizes; and
23 determining a desired ECC capability.

24 19. The method of claim 18, further comprising:
25 listing all frequently-used, multiple packet transactions;
26 determining a smallest transaction from the list; and
27 determining if an alternative ECC may be used with the smallest transaction.

28 20. The method of claim 19, further comprising:
29 if the alternative ECC may be used, encoding all listed transactions using the
30 alternative ECC; and
31 if the alternative ECC may not be used:

- 1 removing from the list, all transactions whose size equals that of the
- 2 smallest transaction, and
- 3 repeating the process until the list is empty.